**Guía FP**

**Tipo de dato:**  Define el tamaño de los contenedores

**Lenguaje de Programación:** Manera en que las computadoras entienden nuestro lenguaje

* **Bajo nivel:** Difíciles de aprender pero eficientes, usan binario **(**Ensamblador)
* **Alto nivel:** Utiliza palabras comunes para los humanos

**Clasificación de los Lenguajes:**

**Procedular:** Instrucción por instrucción, basado en procedimiento y orden secuencial

**Orientado a objetos:** Usan entidades del mundo parta pasarlas a programación, usando paradigmas de atributos y funcionalidad

**Funcional:** Son más matemáticos y lógicos

**Lógico:** Son más matemáticos y lógicos

**1. Tipos de Datos**

Los tipos de datos son la base para almacenar información en memoria. En pseudocódigo y C, se clasifican en **simples** (usan una celda de memoria) y **estructurados** (usan múltiples celdas). Principales simples:

* **Numéricos**:
  + **Enteros**: Números sin decimales, positivos o negativos.
  + **Decimales (o no enteros)**: Números con punto decimal. se declaran como float o double (double tiene más precisión).
* **Lógicos o Booleanos**: Solo dos valores: Verdadero (V, True) o Falso (F, False).
* **Caracteres**: Símbolos alfanuméricos (letras, dígitos, símbolos). Siempre entre comillas simples: 'a', '1', '+'.

**Estructurado**: Utilizamos celdas, depende de lo que se almacene, utilizamos comillas (“”) para formar cadenas de caracteres.

**Memoria y Almacenamiento**: La memoria se divide en celdas con direcciones únicas (0 a N). Datos simples usan una celda.

**2. Identificadores**

Los identificadores son nombres que se utilizan para nombrar variables, funciones, etiquetas o cualquier otro elemento del programa. Son fundamentales para distinguir y acceder a los elementos definidos por el programador.

* **Definición**: Un identificador es una secuencia de caracteres (letras, dígitos y guiones bajos) que representa un nombre único en el programa. Sirve como referencia a variables, funciones u otros objetos.
* **Reglas de Nomenclatura**:
* **Variables**:
  + Deben comenzar con una letra (a-z, A-Z) o un guion bajo (\_).
  + Pueden incluir letras, dígitos (0-9) y guiones bajos, pero no caracteres especiales (ej.: @, #, $, %) ni espacios.
  + No tienen límite de longitud, pero solo los primeros 8 caracteres son significativos en algunos compiladores.
  + Son sensibles a mayúsculas y minúsculas (ej.: miVariable ≠ mivariable).
  + No pueden ser palabras reservadas del lenguaje (ej.: int, if, else, while).

**3. Expresiones y Operadores en C**

Las expresiones combinan variables, constantes y operadores para producir un valor. Toda expresión que se construya debe cumplir:

**Operando 1 Operador Operando 2**

**Operadores Aritméticos** utilizados para construir expresiones aritméticas, se utilizan datos numéricos, operadoeres:

| **Operador** | **Acción** |
| --- | --- |

|  |  |
| --- | --- |
| + | Suma |
| - | Resta |
| \* | Multiplicación |
| / | División (truncada en enteros, ej.: 3/2=1) |
| % | Resto (módulo) |
| ++ | Incremento (pre: ++x evalúa después de sumar; post: x++ evalúa antes) |
| -- | Decremento (similar a ++) |

**Operadores Relacionales** Comparan valores (numéricos, lógicos, caracteres y cadenas):

|  |  |
| --- | --- |
| > | Mayor |
| >= | Mayor o igual |
| < | Menor |
| <= | Menor o igual |
| == | Igual |
| != | Distinto |

* **Operadores Lógicos** (para booleanos):

|  |  |
| --- | --- |
| && | AND (ambos V) |
|  |  |
| ! | NOT (invierte) |

**4. Funciones de Entrada/Salida: printf y scanf en C**

* **printf**: Imprime con formato. Formato: printf("cadena con %especificadores", variables);.
  + Especificadores comunes:

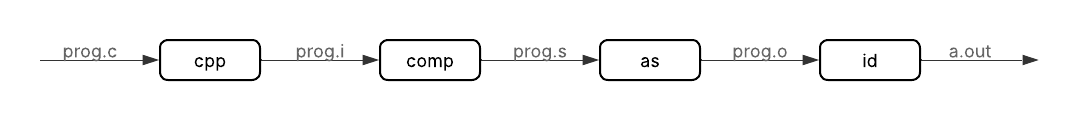
| **Código** | **Formato** |
| --- | --- |

|  |  |
| --- | --- |
| %c | Carácter |
| %d/%i | Entero decimal |
| %f | Flotante |
| %e | Notación científica |
| %s | Cadena |
| %x | Hexadecimal |
| %% | Signo % |

* + Caracteres especiales (escape): \n (nueva línea), \t (tab), \a (beep), " (comillas dobles), \ (barra invertida).
* **scanf**: Lee entrada. Formato: scanf("%especificadores", &variable);. Usa & para dirección de memoria.

**5.Proceso de Compilación en C**

El proceso de compilación en C transforma el código fuente (escrito en lenguaje humano-legible) en un ejecutable (código máquina que la CPU puede correr). Es un proceso multi-etapa, manejado por herramientas como el compilador GCC (GNU Compiler Collection) o similares. No se ejecuta el código directamente; debe compilarse primero debido a que C es un lenguaje compilado (a diferencia de interpretados como Python). Las etapas principales son secuenciales y automáticas en un compilador moderno, pero es útil entenderlas por separado.



* **Etapas del Proceso de Compilación**:
  1. **Preprocesamiento (Preprocessing)**: Limpia basura del código fuente como los comentarios
  2. **Enlazado (Linkiner)**: Importa librerías
  3. **Depurador (Debugger)**: Verifica los errores del programa, línea por línea
  4. **Ensamblado (Assembly)**: el código objeto pasa a lenguaje maquina
* **Comando Típico en GCC**: gcc programa.c -o programa (compila todo en un paso). Opciones comunes: -c (solo hasta objeto), -E (solo preprocesar), -S (hasta ensamblador).
* **Errores Comunes**: Sintaxis (en compilación), enlazado (funciones no definidas), runtime (ejecución, como división por cero).

**6.Comandos Básicos de Linux**

**Navegación y Listado de Archivos:**

* **ls:** Lista contenidos de un directorio (archivos, carpetas). Por defecto, el actual.
  + ls (simple)
  + ls --help (ayuda)
  + ls --color=auto (colorea salida para diferenciar tipos).
* **pwd:** Muestra el directorio actual (ruta completa).
  + pwd → /home/usuario/Documentos.
* **cd:** Cambia de directorio. cd Videos (relativo),
  + cd /home/usuario (absoluto)
  + cd .. (subir nivel)
  + cd - (anterior), cd (al home).

**Gestión de Archivos y Directorios:**

* **cp:** Copia archivos o directorios.
* **rm:** Elimina archivos o directorios.
* **mv:** Mueve o renombra archivos/directórios.
* **mkdir:** Crea directorios/carpetas.
* **touch:** Crea archivos vacíos o actualiza el tiempo de ejecucion.
* **unzip:** Descomprime archivos ZIP.

**Permisos y Ejecución:**

* **chmod**: Cambia permisos de archivos (ejecutable, lectura, etc.). chmod +x script (hacer ejecutable), luego ./script para correr.

**Ayuda y Documentación:**

* **man:** Manual de comandos.
  + man mkdir (manual de mkdir)
* **--help:** Flag general para ayuda rápida.
  + ls --help.
* **whatis:** Descripción corta de un comando.
  + whatis python → "python (1) - an interpreted...".

**Gestión de Sistema y Procesos:**

* **sudo:** Ejecuta como superusuario (root).
  + sudo apt install gimp, sudo cd /root/.
* **shutdown:** Apaga o reinicia el sistema.
  + shutdown now (inmediato),
  + shutdown 20:40 (programado)
  + shutdown -c (cancelar).
* **htop:** Monitor interactivo de procesos (mejor que top).
  + htop (muestra CPU, memoria, procesos).
* **ps:** Lista procesos actuales.
* **kill:** Termina procesos por PID o nombre.
  + kill 533494 (por PID).
  + kill firefox (por nombre).
* **ping:** Prueba conectividad de red.
  + ping google.com
  + ping 8.8.8.8.
* **uname:** Información del sistema.
* **neofetch:** Muestra info del sistema de forma visual.
* **history:** Lista comandos ejecutados previamente.
* **passwd:** Cambia contraseña del usuario.

**Visualización y Búsqueda de Contenido:**

* **cat:** Muestra contenido de archivos.
* **less:** Visualiza archivos paginados (mejor que cat para largos).
* **tail:** Muestra las últimas líneas.
* **head:** Muestra las primeras líneas.
* **grep:** Busca patrones en archivos.
* **wc:** Cuenta líneas, palabras, caracteres.
* **find:** Busca archivos por nombre o tipo.

**Otros Utilitarios:**

* **alias:** Crea atajos para comandos.
* **unalias:** Elimina alias.
* **echo:** Imprime texto o variables.
* **which:** Muestra ruta de un ejecutable.
* **shred**: Borra archivos de forma segura (sobrescribe).
* **whoami**: Muestra el usuario actual
* **wget**: Descarga archivos de la web.
* **vim**: Editor de texto avanzado.
* **exit**: Sale de la terminal o script.